News Classifier

FAKE NEWS DETECTION

By: Amechi Aduba

01 Problem

02 Dataset

03 Preprocessing

04 Data Analysis

05 Models

06 Evaluation

07 Deployment

08 Conclusion

Problem

- Motivation: In current social media and politics, there is a large issue of misinformation spreading to those trying to gain information about the world around them
- The Goal: To determine which news articles are true and false

Dataset

- Source: Kaggle Fake and True news dataset
  - Contains over 21k true articles and 23k false articles
- The columns are title, text subject and date
- But the only ones we will use to process is the text section

Preprocessing

- To preprocess the data I had to:
  - Lowercase the text
  - Ensure punctuation and digit removal
  - Tokenize and remove stopwords
  - Stem the words with the porterstemmer
  - TF-IDF Vectorization capped at 5,000 features
  - MiniLM fine tuning

```
def preprocess_strings(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation + string.digits))
    words = word_tokenize(text)
    words = [word for word in words if word.isalpha()]  # Keep only words
    words = [word for word in words if word not in stop_words]
    words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(words)
```

Data Analysis

- Gather training data on 80/20 split
- Generated a confusion matrix and a classification report
- ROC Curve
- To analyze the data I got the top words in both the true and fake datasets with the stemmer

```
True News Most Common Words:  [('said', 99042), ('trump', 54373), ('u', 41177), ('state', 36647), ('would', 31525), ('reuter',
28404), ('presid', 28386), ('republican', 22114), ('govern', 20220), ('year', 19280), ('hous', 17529), ('new', 16787), ('unit',
16525), ('democrat', 16222), ('also', 15953), ('say', 15948), ('senat', 15707), ('elect', 15528), ('peopl', 15324), ('parti', 1
5005)]
Fake News Most Common Words:  [('trump', 74286), ('said', 31016), ('presid', 27947), ('peopl', 26054), ('one', 23752), ('stat
e', 23721), ('would', 23427), ('u', 22353), ('like', 21412), ('say', 20522), ('clinton', 18652), ('go', 18146), ('time', 1807
0), ('obama', 18070), ('donald', 17219), ('republican', 16041), ('american', 16036), ('also', 15242), ('year', 15201), ('get',
15024)]
```

Model Approaches

- In order to try to accurately predict real and fake news I began with the Multinomial Naive Bayes model
- However after finally deploying the model I determined that the accuracy was not as expected and used Logistic Regression instead
- After further testing I discovered that the Logistic Regression worked poorer than Naive Bayes on real news sources so I boosted it with a pre trained Language Model encoder

How the pre-trained MiniLM encoder works (Model Approaches)

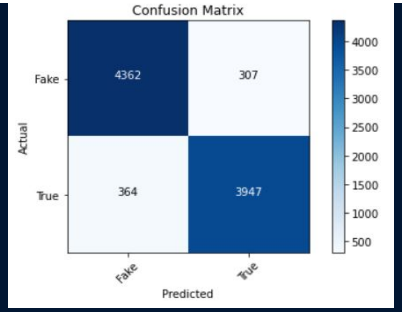
- TF-IDF counts ignore word order and meaning which ended up misclassifying unfamiliar sources.
- If I had used the Full BERT fine-tune it will be accurate but heavy (GPU, long training)
- MiniLM is a light DistilBERT-style encoder that keeps >95 % of BERT's accuracy at 40 % of the size.
- Converts each article into a 384-dimensional sentence embedding that captures context, tone, and semantics.
- No training needed for the encoder, only encoding

Results

Naive Bayes

Classification Report:

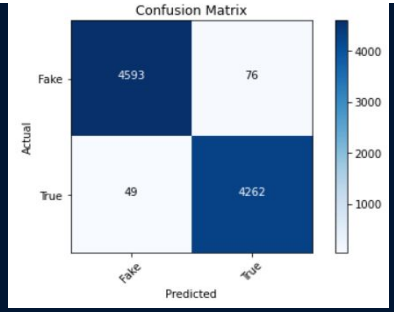
	precision	recall	f1-score	support
0	0.92	0.93	0.93	4669
1	0.93	0.92	0.92	4311
accuracy			0.93	8980
macro avg	0.93	0.92	0.93	8980
weighted avg	0.93	0.93	0.93	8980



Logistic Regression

Classification Report for Logistic Regression:

	precision	recall	f1-score	support
0	0.99	0.98	0.99	4669
1	0.98	0.99	0.99	4311
accuracy			0.99	8980
macro avg	0.99	0.99	0.99	8980
weighted avg	0.99	0.99	0.99	8980



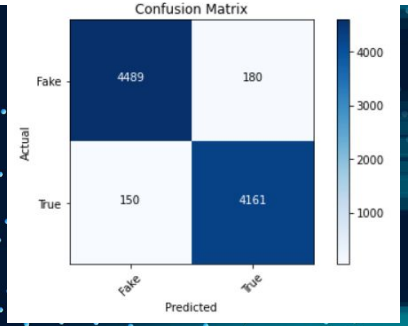
MiniLM + LogReg

Confusion Matrix (MiniLM + LR):

```
[[4489  180]
 [ 150 4161]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	4669
1	0.96	0.97	0.96	4311
accuracy			0.96	8980
macro avg	0.96	0.96	0.96	8980
weighted avg	0.96	0.96	0.96	8980



Deployment

http://127.0.0.1:7864/

- Successfully built a fake news classifier using two models: Naive Bayes and Logistic Regression
- Achieved ~93% accuracy with Naive Bayes and ~99% with Logistic Regression, and 96 with contextual Log Reg.
- Deployed a functional web interface with Gradio for live predictions
- Gave users a choice between each model

Conclusion

- Takeaways:
  - Text preprocessing significantly impacts performance
  - Despite their differences in accuracies, Naive Bayes seemed to perform better on real news sources.
  - Encoding and Boosting a model can improve it for different contexts
- Limitations:
  - Dataset is from a specific time and political domain (most of it is from 2015–2017)
  - Could benefit from testing on newer data or multilingual support

Next Steps:

- Highlight the words that are influencing predictions
- Expand to classify articles as satire, clickbait, or biased news
- Create a real-time scraping and classification model to maximize efficiency

Sources

- https://www.analyticsvidhya.com/blog/2023/02/tackling-fake-news-with-machine-learning/#:~:text=The%20dataset%20we%20used%20for,an%20accuracy%20of%20over%2099
- https://www.geeksforgeeks.org/seaborn-heatmap-a-comprehensive-guide/
- https://www.geeksforgeeks.org/naive-bayes-classifiers/
- https://www.geeksforgeeks.org/understanding-logistic-regression/
- https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
- https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset